

# A Simple Algorithm for Maximum Margin Classification, Revisited

Sariel Har-Peled\*

July 7, 2015

In this note, we revisit the algorithm of Har-Peled *et al.* [HRZ07] for computing a linear maximum margin classifier. Our presentation is self contained, and the algorithm itself is slightly simpler than the original algorithm. The algorithm itself is a simple Perceptron like iterative algorithm. For more details and background, the reader is referred to the original paper.

## 1. Active learning, sparsity and large margin

Let  $P$  be a point set of  $n$  points in  $\mathbb{R}^d$ . Every point has a label/color (say *black* or *white*), but we do not know the labels. In particular, let  $\mathbf{B}$  and  $\mathbf{W}$  be the set of black and white points in  $P$ . Furthermore, let  $\Delta = \text{diam}(P)$ , and assume that there exist two parallel hyperplanes  $h, h'$  in distance  $\gamma$  from each other, such that the slab between  $h$  and  $h'$  does not contain an point of  $P$ , and the points of  $\mathbf{B}$  are on one side of this slab, and the points of  $\mathbf{W}$  are on the other side. The quantity  $\gamma$  is the *margin* of  $P$ .

A somewhat more convenient way to handle such slabs, is to consider two points  $\mathbf{b}$  and  $\mathbf{w}$  in  $\mathbb{R}^d$ . Let  $\text{slab}(\mathbf{b}, \mathbf{w})$  be the region of points in  $\mathbb{R}^d$ , such that their projection onto the line spanned by  $\mathbf{b}$  and  $\mathbf{w}$  is contained in the open segment  $\mathbf{bw}$ . We use  $(1 - \varepsilon)\text{slab}(\mathbf{b}, \mathbf{w})$  to denote the slab formed from  $\text{slab}(\mathbf{b}, \mathbf{w})$  by shrinking it by a factor of  $(1 - \varepsilon)$  around its middle hyperplane. Formally, it is defined as  $(1 - \varepsilon)\text{slab}(\mathbf{b}, \mathbf{w}) = \text{slab}(\mathbf{b}', \mathbf{w}')$ , where  $\mathbf{b}' = (1 - \varepsilon/2)\mathbf{b} + (\varepsilon/2)\mathbf{w}$  and  $\mathbf{w}' = (\varepsilon/2)\mathbf{b} + (1 - \varepsilon/2)\mathbf{w}$ .

In the following, we assume have an access to a *labeling oracle* that can return the label of a specific query point. Similarly, we assume access to a *counterexample oracle*, such that given a slab that does not contain any points of  $P$  in its interior, and supposedly separates the points of  $P$  into  $\mathbf{B}$  and  $\mathbf{W}$ , it returns a point that is mislabeled by this classifier (i.e., slab) if such a point exists.

Conceptually, asking queries from the oracles is quite expensive, and the algorithm tries to minimize the number of such queries.

**The algorithm.** Assume there are two points  $\mathbf{b}_1 \in \mathbf{B}$  and  $\mathbf{w}_1 \in \mathbf{W}$ . For  $i > 0$ , in the  $i$ th iteration, the algorithm considers the slab  $S_i = (1 - \varepsilon)\text{slab}(\mathbf{b}_i, \mathbf{w}_i)$ . There are two possibilities:

- (A) If the slab  $S_i$  contains no points of  $P$ , then the algorithm uses the counterexample oracle to check if it is done – that is, all the points are classified correctly. Otherwise, a badly classified point  $p_i$  was returned.

---

\*Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; <http://sarielhp.org/>. Work on this paper was partially supported by a NSF AF awards CCF-1421231, and CCF-1217462.

(B) The  $S_i$  contains some points of  $P$ , and let  $p_i$  be the closest point to the middle hyperplane of the slab  $S_i$ . The algorithm uses the labeling oracle to get the label of  $p_i$ .

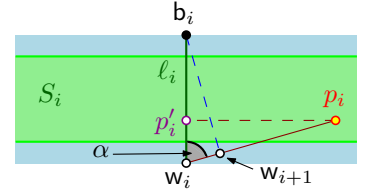
Assume that the label of  $p_i$  is white. Then, the algorithm set  $\mathbf{w}_{i+1}$  be the projection of  $\mathbf{b}_i$  to  $\mathbf{w}_i p_i$ , and  $\mathbf{b}_{i+1} = \mathbf{b}_i$  (the case that  $p_i$  is black is handled in a symmetric fashion).

**Lemma 1.1** ([HRZ07]). *Let  $P$  be a set of points in  $\mathbb{R}^d$ , with diameter  $\Delta$ . Assume there is an unknown partition of  $P$  into two (unknown) point sets  $\mathbf{B}$  and  $\mathbf{W}$ , of white and black points, respectively, and this partition has margin  $\gamma$ . Furthermore, we are given an access to a labeling and counterexample oracles. Finally, there are two given points  $\mathbf{b}_1 \in \mathbf{B}$  and  $\mathbf{w}_1 \in \mathbf{W}$ .*

*Then, for any  $\varepsilon > 0$ , one can compute using an iterative algorithm, in  $I = O((\Delta/\gamma)^2/\varepsilon^2)$  iterations and in  $O(I \text{dn})$  time, a slab of width  $\geq (1 - \varepsilon)\gamma$  that separates  $\mathbf{B}$  from  $\mathbf{W}$ . This algorithm performs  $I$  calls to the labeling/counterexample oracles.*

*Proof:* Our purpose is to analyze the number of iterations of this algorithm till it terminates. So, let  $\ell_i = \|\mathbf{b}_i - \mathbf{w}_i\|$ . Clearly,  $\Delta \geq \ell_0 \geq \ell_1 \geq \dots \geq \gamma$ , the last step follows as  $\mathbf{b}_i \in \mathcal{C}_{\mathbf{B}}$  and  $\mathbf{w}_i \in \mathcal{C}_{\mathbf{W}}$ , and the distance  $d(\mathcal{C}_{\mathbf{B}}, \mathcal{C}_{\mathbf{W}}) \geq \gamma$ , where  $d(X, Y) = \min_{x \in X} \min_{y \in Y} \|x - y\|$ .

Let  $p'_i$  be the projection of  $p_i$  to the line spanned by  $\mathbf{w}_i \mathbf{b}_i$ . Observe that if  $p_i \in S_i$  then  $\|p'_i - \mathbf{w}_i\| \geq \varepsilon \ell_i / 2$ . Formally, the points  $\mathbf{w}_i$  breaks the line spanned by  $\mathbf{w}_i$  and  $\mathbf{b}_i$  into two parts, and  $\mathbf{b}_i$  and  $p'_i$  are on the same side, and  $p'_i$  is distance at least  $\ell_i/2$  away from  $\mathbf{w}_i$  along this ray. Observe that if case (B) above happened, then  $p_i$  is not inside  $S_i$ , and this distance is significantly larger.



Setting  $\alpha = \angle p_i \mathbf{w}_i \mathbf{b}_i$ , we have  $\cos \alpha = \frac{\|p'_i - \mathbf{w}_i\|}{\|\mathbf{w}_i - p_i\|} \geq \frac{\varepsilon \ell_i / 2}{\Delta}$ . As such, we have

$$\ell_{i+1} = \ell_i \sin \alpha \leq \ell_i \sqrt{1 - \left(\frac{\varepsilon \ell_i}{2\Delta}\right)^2} \leq \left(1 - \left(\frac{\varepsilon \ell_i}{4\Delta}\right)^2\right) \ell_i. \quad (1.1)$$

We have that  $\ell_{i+k} \leq \ell_i/2$ , for  $k = \left\lceil 64\Delta^2/(\varepsilon \ell_i)^2 \right\rceil$ . Indeed, if  $\ell_{i+k} > \ell_i/2$ , then

$$\ell_{i+k} \leq \ell_i \prod_{j=0}^{k-1} \left(1 - \left(\frac{\varepsilon \ell_{i+j}}{4\Delta}\right)^2\right) \leq \ell_i \prod_{j=0}^{k-1} \left(1 - \left(\frac{\varepsilon \ell_{i+k}}{4\Delta}\right)^2\right) \leq \ell_i \exp\left(-k \left(\frac{\varepsilon \ell_{i+k}}{4\Delta}\right)^2\right) \quad (1.2)$$

$$\leq \ell_i \exp\left(-k \left(\frac{\varepsilon \ell_i}{8\Delta}\right)^2\right) \leq \ell_i \exp\left(-k \left(\frac{\varepsilon \ell_i}{8\Delta}\right)^2\right) \leq \frac{\ell_i}{e}, \quad (1.3)$$

which is a contradiction.

In particular, the  $j$ th epoch of the algorithm are the iterations where  $\ell_i \in [\Delta/2^{j-1}, \Delta/2^j]$ . Namely, during an epoch the width of the current slab shrinks by a factor of two. By Eq. (1.3), the  $j$ th epoch lasts  $n_j = O((2^j/\varepsilon)^2)$  iterations. As such, the total number of iterations  $\sum_j n_j$  is dominated by the last epoch, that starts (roughly) when  $\ell_i \leq 2\gamma$ , and end when it hits  $\gamma$ . This last epoch takes  $O(\Delta^2/(\varepsilon\gamma)^2)$  iterations, which also bounds the total number of iterations.  $\blacksquare$

**Remark.** (A) if the data is already labeled, then the algorithm of Lemma 1.1 can be implemented directly resulting in the same running time as stated. This algorithm approximates the maximum margin

classifier to the data. Specifically, the above algorithm  $(1 + \varepsilon)$ -approximates the distance  $d(\mathbf{B}, \mathbf{W})$ , and it can be interpreted as an approximation algorithm for the associated quadratic program.

(B) One can implement the counterexample oracle, by sampling enough labels, and using the labeling oracle. This introduces a certain level of error. See [HRZ07] for details.

## References

- [HRZ07] [S. Har-Peled](#), D. Roth, and D. Zimak. Maximum margin coresets for active and noise tolerant learning. In *IJCAI*, pages 836–841, 2007.